IBM Research

SOA as an Architectural Pattern

Best Practices in Software Architecture

Grady Booch
Free Radical

What's An Enterprise?

- **A namable entity that delivers some product or service with some measureable value**
  - An enterprise has boundaries (but is rarely completely isolated)
  - An enterprise does something (for some set of stakeholders)
  - An enterprise's activities can be measured (by the market or by less tangible measures)

What Is Not An Enterprise?

- **An enterprise is not**
  - A product or service
  - A project
  - A family of interconnected projects
  - An architecture of these products and services
  - The human organization
- **Although these things are all elements of an enterprise**

The Living Enterprise

- **All vibrant enterprises are quite dynamic**
- **Software-intensive systems are a primary mechanism used by enterprises to carry out their mission**
  - May be the very soul of the enterprise (eBay, Amazon, BoF)
  - Might be the product or service itself (Intuit, Cisco)
  - Might be the touch point of its interaction with the world (NRO)
- **All well-structured systems are full of patterns**

EA Is Not TA

- **Although the two share the noun "architecture" they are different things.**
  - EA attends to the architecture of a business that uses technology
  - TA attends to the architecture of the software-intensive systems that support the business.
- **Each domain - that of the business and that of the system - have fundamentally different stakeholders with different perspectives and different viewpoints.**

What is a Pattern?

- **Patterns describe a solution to a recurring problem.**
  - Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution. (Christopher Alexander)
  - A pattern may be discovered in the collaboration of people or within workflows and procedures, and so on
  - Typically someone identifies the recurrence of the problem and the solution and formally documents the participants, their interactions, and the context
  - Some patterns are best captured as guidance and persist in the form of documentation such as RUP or GS Method. Other patterns may be implemented with automation
  - Patterns may live at different levels of systems; such as from business processes to the deployed bits

What is a Framework?

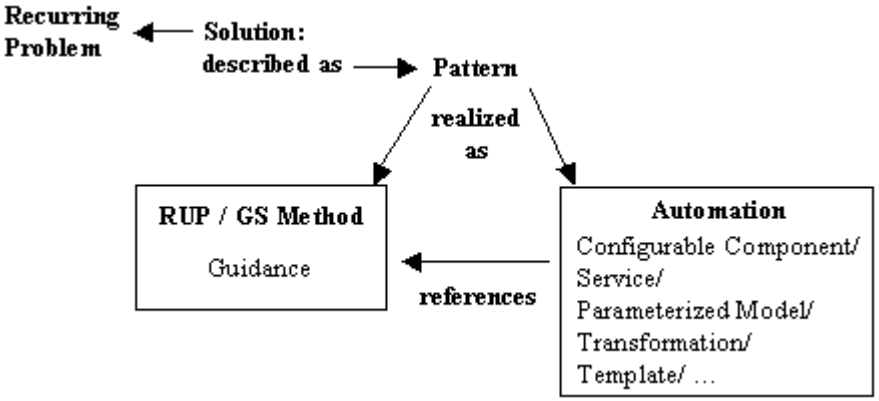
- **Partially implemented application built from patterns.**
- **May use a pattern solution to implement a framework**

Pattern Value

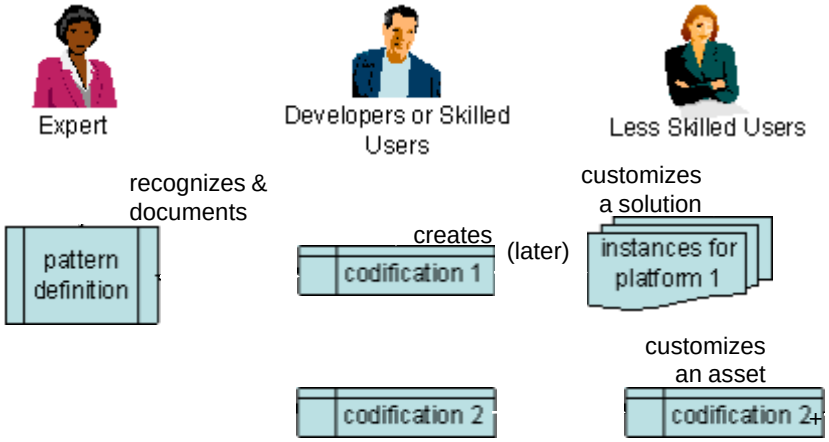
- **Organizations seek improved quality and productivity, consistency across development efforts, communicating and establishing best practices, improved governance and architectural control**
- **Using patterns in our daily activities reduces the mundane, improves productivity and quality, and leaves more cycles for the creative juices to be applied to competitive differentiators**
- **Today's patterns often become tomorrow’s infrastructure**

Implementing Patterns

- **Patterns may be realized as process guidance or may be codified**



The solution described by a pattern may be implemented in a component, or across several components, or in a service. There are various codifications of patterns.



•Configurable Component: parameterized software component implementing one or more patterns; may be published as a service

•Transformation: implements a pattern through actions which modify model elements and other artifacts

•Template: contains sections marked for substitution with parameters supplied by the user

•Parameterized Model: a special kind of template, this describes a pattern to which model elements may be mapped to “apply” the pattern

The codified pattern may be further customized by less skilled users.

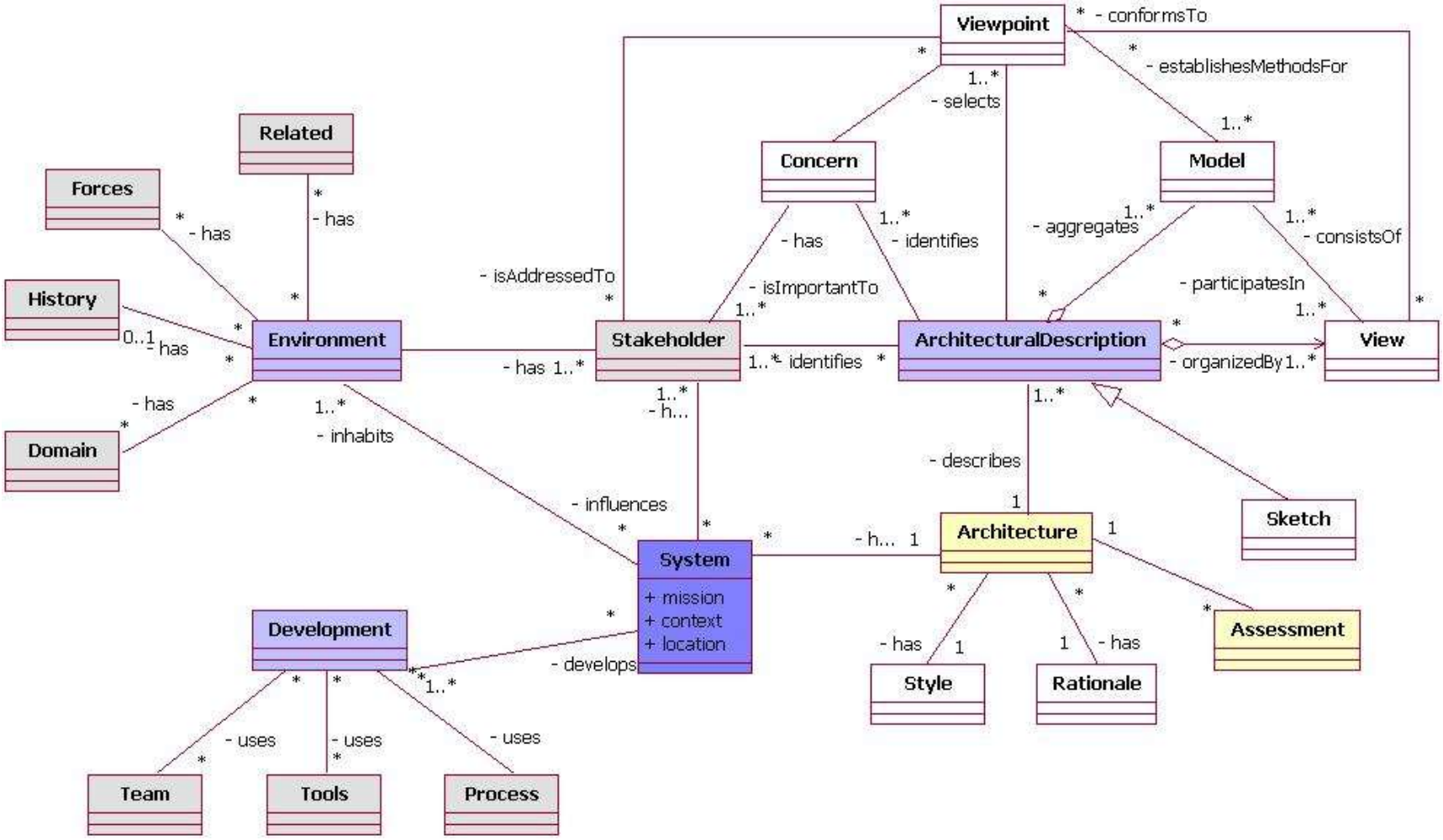
Narrowing Our Scope Of Interest

- **We largely care about the creation, development, deployment, evolution, operation, and support of these software-intensive systems**
- **The operative phrase here is "software-intensive system"**
  - It's about software *and* hardware *and* the social elements
  - The most important artifact is the raw, running naked code that runs on hardware and interacts with humans
  - All other artifacts are secondary, but nonetheless they are still critical, for they help the enterprise deliver the right system at the right time to the right stakeholders with the right balance of cost and value.

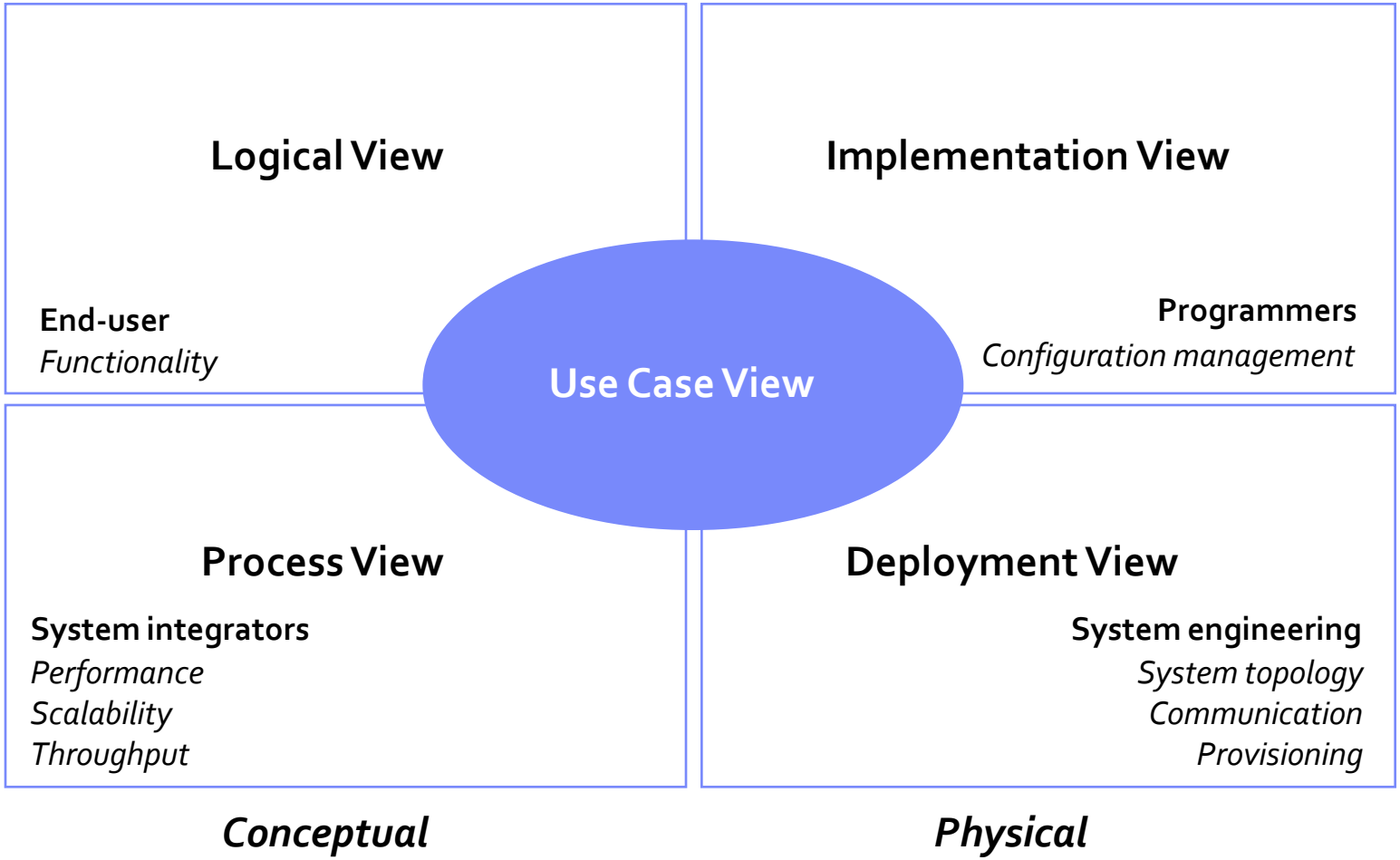
There Are Some Known Knowns

- **All architecture is design; not all design is architecture. A system's architecture is defined by its significant design decisions (where "significant" is measured by the cost of change)**
- **Most architectures are accidental; some are intentional**
- **Every software-intensive system has an architecture, forged from the hundreds of thousands of small decisions made every day**
- **The code is the truth, but not the whole truth: most architectural information is preserved in tribal memory**

Architecture metamodel



Representing software architecture



Kruchten, "The 4+1 Model View"

Misconceptions About Architecture

- **Architecture is just paper**
- **Architecture and design are the same things**
- **Architecture and infrastructure are the same things**
- ***<my favorite technology>* is the architecture**
- **A good architecture is the work of a single architect**
- **Architecture is simply structure**
- **Architecture can be represented in a single blueprint**
- **System architecture precedes software architecture**
- **Architecture cannot be measured or validated**
- **Architecture is a science**
- **Architecture is an art**

Architectural Patterns

- **Event-driven**
- **Blackboard**
- **Pipe and filter**
- **Semantic core**
- **Message passing**
- **…**

SOA as an Evolutionary Development

- **CICS et al are essentially message-passing technologies**
- **From the 90s to the presence, considerable investment has been made in web infrastructures**
- **One had to pass messages across firewalls, using web protocols**
  - RM-ODP (1996-1998)
  - COM -> SOAP (1999)
  - WSDL (2007)

SOA as an Architectural Pattern

- **Message-passing appears to be a fundamental architectural style**
  - SOA is in effect a specific manifestation
- **Services might be a big S or a little s**
  - Web services (big S)
  - Other kinds of services (RPC, CICS, etc)

SOA as a Generative Pattern

- **If you consider SOA to be a message-passing style, then several decisions follow**
  - Granularity of services
  - Separation of concern among services
  - Semantics of the messages themselves
  - Processes to continuously refactor these services and messages

Fundamentals

- **Development takes place at two levels: architecture and implementation.**
  - Both are ongoing, and they interact with each other strongly. New implementations suggest architectural changes. Architectural changes usually require radical changes to the implementation.

Coplien, *Organizational Patterns of Agile Development*, p. 332

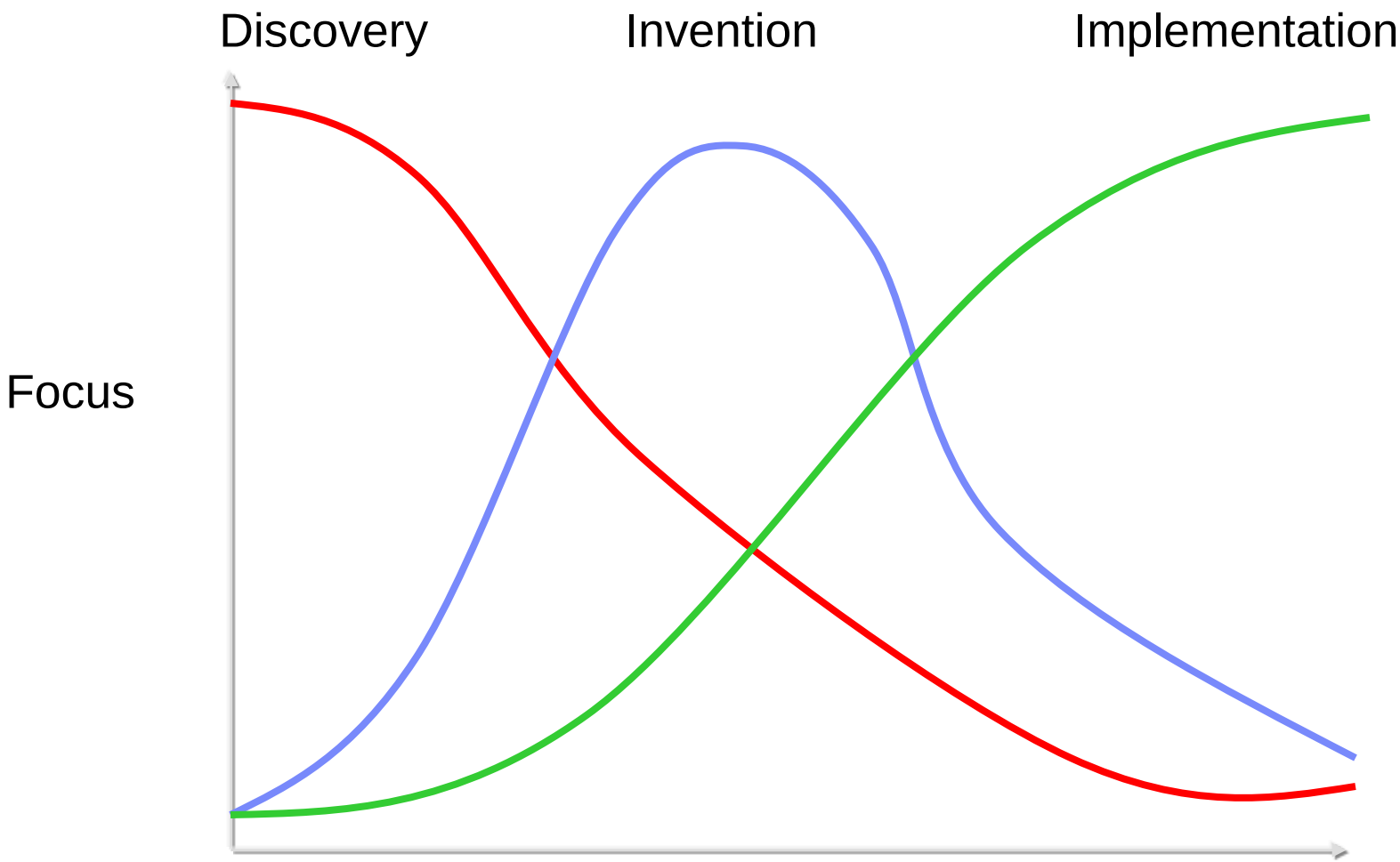
What Pain Do You Feel?

- **How do we attend to new requirements without being saddled by our legacy (but at the same time not compromising that legacy?)**
- **How do we integrate new technology into our existing code base?**
- **How do we integrate our existing systems to extract greater value from the whole?**
- **How do we increase our agility in response to the market while simultaneously improving efficiency and quality yet also reducing costs?**
- **How do we attend to assets introduced through acquisition?**
- **How do use software to improve market efficiency through the creation of dominant product lines?**
- **How do we attend to a continuously refreshed stakeholder community, a globally and temporally distributed development team, and inevitable leakage/loss of institutional memory?**
- **While doing all this, how do we continue to innovate?**

An Observation

- **While these points of pain are legion, a common thread that weaves though them is that of architecture**
  - Every software-intensive system has one
  - Most are accidental, a few are intentional
  - A considerable amount of architectural knowledge lies in tribal memory
- **The presence of a reasonably well understood, syndicated, and governed architecture has a positive impact upon each of these points of pain**

Focus over time



Bran Selic

The Enterprise Architecture Lifecycle

- **In my experience**
  - All hyperproductive organizations tend to have a lifecycle that involves the growth of a system's architecture through the incremental and iterative release of testable executables.
- **Not one lifecycle, but many**
  - Different stages of execution, maturation, and quality
  - Harmony, resonance, and cacaphony

Best Practices For Software-Intensive Systems

- **Architecture-as-artifact is a manifestation of technical intellectual property and thus serves as an artifact of control involving**
  - Active yet flexible budgeting of resources
  - Checks and balances for the co-evolution of architecture and implementation
  - Accountability for technical decisions
  - Hedges for the future
  - Diversification for the future
  - Appropriate measurements and incentives
  - Cost controls
  - Economics of scale via patterns
  - Actively attacking risk

Things You Can Do With Old Software

- **Give it away**
- **Ignore it**
- **Put it on life support**
- **Rewrite it**
- **Harvest from it**
- **Wrap it up**
- **Transform it**
- **Preserve it**

Things You Can Do With An Architecture

- **Reason about its transformation**
  - Evolution, rapid reaction, modernization, merging, acquisition, divestiture
- **Asset identification and repurposing**
  - Product line
  - Strategic thrust

Fundamentals Never Go Out Of Style

- **Crisp abstractions**
- **Clear separation of concerns**
- **Balanced distribution of responsibilities**
- **Simplicity**

- **Grow a system through the iterative and incremental release of an executable architecture**